

A COMPARISON OF METAHEURISTICS ALGORITHMS FOR COMBINATORIAL OPTIMIZATION PROBLEMS. APPLICATION TO PHASE BALANCING IN ELECTRIC DISTRIBUTION SYSTEMS

G.A. SCHWEICKARDT[†], V. MIRANDA[‡] and G. WIMAN^{*}

[†] Instituto de Economía Energética/CONICET – Fundación Bariloche – Av. Bustillo km 9,500 – Centro Atómico Bariloche – Pabellón 7, Argentina. gustavoschweickardt@conicet.gov.ar

[‡] INESC Porto – Instituto de Engenharia de Sistemas e Computadores do Porto and FEUP – Fac. Engenharia Univ. Porto P. República 93 - 4050 Porto, Portugal. vmiranda@inescporto.pt

^{*} Universidad Tecnológica Nacional – FR BA, Ext. Bariloche – Fanny T. de Newbery 111, Bariloche, Argentina gwiman@utn.frba.edu.ar

Abstract— Metaheuristics Algorithms are widely recognized as one of most practical approaches for Combinatorial Optimization Problems. This paper presents a comparison between two metaheuristics to solve a problem of Phase Balancing in Low Voltage Electric Distribution Systems. Among the most representative mono-objective metaheuristics, was selected Simulated Annealing, to compare with a different metaheuristic approach: Evolutionary Particle Swarm Optimization. In this work, both of them are extended to fuzzy domain to modeling a multi-objective optimization, by mean of a fuzzy fitness function. A simulation on a real system is presented, and advantages of Swarm approach are evidenced.

Keywords— Metaheuristic Algorithm; Swarm Intelligence; Fuzzy Sets; Electric Distribution; Phase Balancing.

I. INTRODUCTION

Metaheuristics Algorithms are widely recognized as one of more practical and successful approaches to solve combinatorial problems. However, the original formulations have been oriented to mono-objective optimizations. Many proposal of extensions to multi-objective domain have been established, but each formulation has showed particular advantages and limitations, in general, over certain kind of problems. Does not exist the “best multiobjective metaheuristic algorithm”, but some algorithms are more appropriate for certain problems. Such is the case of Phase Balancing in Low Voltage Electric Distribution Systems (LVEDS), when a classic programming approach is not addressed to solve it. The balance is referred to the loads in the feeders of a LV network in an EDS. The classic approach, has demonstrated major limitations, as it will be discussed. For this reason, a metaheuristic approach is an alternative that may produce very good results.

This work is organized as follows: in the section II.A the problem of Phase Balancing is presented. It describe the no desired effects that produce an elevate unbalance degree in the loads of a low voltage (LV) feeder. In section II.B are described the principles of

two mentioned metaheuristics: Simulated annealing (SA) and Evolutionary Particle Swarm Optimization (EPSO). In section II.C, an introduction of the static fuzzy decision is presented, and the extension of the SA and EPSO algorithms to fuzzy domain, by mean of a fuzzy fitness function, is proposed for solve multi-objective optimizations. Two models, designed as FSA and FEPSO, specialized to solve the Phase Balancing Multi-objective Problem, are obtained. Lastly, in the section II.D, a simulation on a real LV feeders system is presented and the results, obtained from FSA and FEPSO, are compared. The conclusions are presented in the section III.

II. METHODS

A. The Problem of Phase Balancing

The most LV networks of an EDS are three-phase systems, physically defined by feeders with three conductors, one per phase (feeders system). If all loads of each conductor-phase were three-phase (with the same value in each phase) then the system would be balanced. However, feeders loads in a LV network, for low incomes residential areas, are commonly single-phase. Original feeders system design, depends on accuracy of the given load data and, even maximizing this accuracy, there will always be certain unbalance degree, due to single-phase loads. A high unbalance degree, produces high voltage drops, high power and energy losses and low reliability. For this reason, such degree must be as low as possible. For the purpose of this paper, LV feeders system will have only single-phase loads. A formulation to the general problem of Phase Balancing, in this context, can be expressed as follows:

$$\text{Min } \{ \text{Loss}_T; I(\Delta u) \}; |I_{[o]}|_f \} \quad (1)$$

Subject to:

$$|I_{[R]}|_f \leq I_{Max} \quad (2)$$

$$|I_{[S]}|_f \leq I_{Max} \quad (3)$$

$$|I_{[T]}|_f \leq I_{Max} \quad (4)$$

where the subindex f , refers to the output of substation connected to the principal feeder of the system; Loss_T are the total active power loss of system and $I(\Delta u)$ is an index that depends on voltage drops.

$I_{[o]f}$ (homopolar component) satisfy the equation:

$$I_{[R]f} + I_{[S]f} + I_{[T]f} = 3 \times I_{[o]f} \quad (5)$$

If the system is balanced, then $|I_{[o]f}| = 0$.

The sub index $[R]$, $[S]$ and $[T]$, refers to each phase of system. In addition, three constraints are imposed: the intensities in each phase at the output, must be less than the phase line capacity, I_{Max} , Eq. (2), (3) and (4). The problem can be seen as a set of swapping single phase loads or a load assignment to lines. For example, a single phase load can only be assigned to either phase $[R]$, $[S]$ or $[T]$ (see Fig. 1). This assignment should be executed until the objectives (1) are satisfied.

In the Fig. 1, the line-dotted rectangles represent nodes in which groups of single phase loads are connected (i.e. residential customers). Swaping the phases (double-arrow curves), the objectives functions (1) are evaluated by mean of a three-phase load flow. P_i is the active power and Q_i is a reactive power connected at the node i . Phase Balancing has several significant benefits, such as improving power quality and reliability, and the utilization factor. More details are presented in the section II.D.

This problem is clearly combinatorial: if there are 3 phases and n loads that can be swapping, then the number of states of search space for the solution, is 3^n . In the reference (Zhu *et al.*, 1998) is proposed to solve it a model based in Linear Mixed-Integer Programming, but it exhibit significant limitations, such as: a) the problem is not linear, and the linear formulation is valid if the current of each individual load is constant. This situation does not occurs in practice; b) the model requires a convex set of parameters, of summatory 1, as subjective weights for each node to weigh the importance of its unbalance degrees. This simple formulation to Multi-Objective Linear Optimization, is a poor approach to search a global minimum in the objectives functions (1).

A metaheuristic approach, perform a better search in the solutions space, because it not require to know, essentially, the characteristic of each objective function.

B. The Metaheuristics SA and EPSO

B.1. Simulated Annealing (SA)

The concept of Simulated Annealing in combinatorial optimization was introduced by Kirkpatrick *et al.* (1983). SA appears like a flexible metaheuristic that is an adequate tool to solve a great number of combinatorial optimization problems. It is motivated by an analogy to annealing in solids and the idea comes from a paper published by Metropolis *et al.* (1953). The Metropolis algorithm simulated the cooling of material in a heat bath. This is the process know as annealing. It consists on two steps: a) the temperature is raised to a state of maximum energy and b) the temperature is slowly lowered until a minimum energy state, equivalent to thermal equilibrium, is reached. The structural properties of the system, depends on the rate of cooling. If it is cooled slowly enough, large crystals will be formed. However, if the system is cooled quickly, the crystals will contain imperfections. Metropolis's algorithm simulated the material as a system of particles.

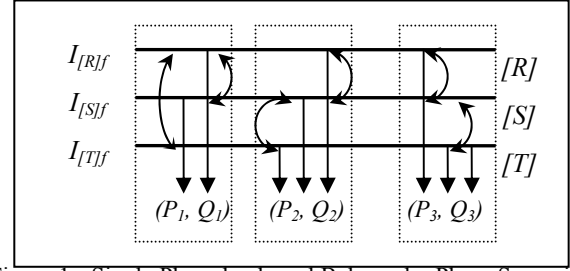


Figure 1 - Single Phase loads and Balance by Phase Swapping in some loads

In order to more clearly explain the SA metaheuristic, is possible present an analogy between a physical system, with a large number of particles, and a combinatorial problem. This analogy can be stated as follows:

- The **solutions** of combinatorial problem are equivalent to the physical system **states**;
- The **attributes** of solutions are equivalent to the **energy** of different states;
- The **control parameter** in the combinatorial problem is equivalent to the **temperature** of physical system.

The evolution of the solution algorithm is simulated using probabilistic sampling techniques, supported by successive generation of states. This process begins with an initial state, i , evaluated by an energy function, $E(i)$. After generating and analyzing a second state, j , $E(j)$, it is performed an acceptance test. The acceptance of this new solution, j , depends on a probability computed by:

$$p(\text{accept } j) = \begin{cases} 1; & \text{if } E(j) \leq E(i) \\ e^{\frac{E(j)-E(i)}{c}}; & \text{if } E(j) \geq E(i) \end{cases} \quad (6)$$

where c is a positive real number, $c = k_B \times T$; k_B is a constant (called Boltzman constant, in Metropolis's Algorithm) and T is a temperature of system.

A procedure is defined, in pseudo-code, as follows:
Minimize $f(i)$ for $i \in S \rightarrow \text{Search Space}$

Begin Procedure SA

1. **set** starting point $i = i_0$
2. **set** starting temperature $T = T_0$ and cooling rate: $0 < \alpha < 1$;
3. **set** N_T (number of trials per temperature);
4. **while** stopping condition is not satisfied **do**
5. **for** $k \leftarrow 1$ to N_T **do**
6. generate trial point j from S_i using $q(i, j)$;
7. accept j with probabylity $p(\text{accept } j)$ (eq. 6);
8. **end for**
9. reduce temperature by $T \leftarrow T \times \alpha$;
10. **end while**

End Procedure SA

S_i is a Neighborhood of solution i : is a set of discrete points j satisfying $j \in S_i \Leftrightarrow i \in S_j$. The generation function of S_i is $q(i, j)$ defined externally.

B.2. Evolutionary Particle Swarm Optimization (EPSO)

The metaheuristic EPSO, is built over the concepts of Evolution Strategies (ES) and Swarm Intelligence (SI).

Under the name of Evolution Strategies and Evolutionary Programming (EP) a number of models to solve combinatorial optimization problems have been developed, that rely on Darwinist selection to promote progress towards the (unknown) optimum. This selection procedure may rely on pure stochastic basis or have a deterministic flavor, but, at the end, the general principle of the “survival of the fitness” remains.

On the other hand, the Swarm Intelligence is adopted by Particle Swarm Optimization metaheuristic, (PSO) (Kennedy and Eberhart, 1995) that rely on a different concept. Mimicking zoological behavior, imitating the collective or social behavior of animals swarms, flocks or schools, a set of particles evolves in the search space motivated by three factors, called *habit*, *memory* and *cooperation*. The first factor impels a particle to follow a path along its previous movement direction. It is frequently called the *inertia* factor. The second factor, influences the particle to come back on its steps (i.e., to tend to go back to the best position it found during its life). The third factor (vinculated to information exchange), induces the particle to move closer to the best point in the space, found by the collective of all particles in its family group. Analogy between Particle Swarm and Combinatorial Problem is easy to see, establishing a correspondence between the position of particle and a solution in the search space.

Before describe the EPSO strategy, a brief review of Classical PSO metaheuristic is presented.

Particle Swarm Optimization (PSO)

In the Classical PSO, one must have, at a given iteration, a set of solutions or alternatives called “particles”. From one iteration to the following, each particle, X_i , moves according to a rule that depends on the three factors described (*habit*, *memory* and *cooperation*). In addition, each particle of swarm keep the record of the best point found in its past life, b_i , and the record of the current global best point found by the swarm, b_G . Then, the PSO Movement Rule, states that (X and V are vectors):

$$X_i^{New} = X_i + V_i^{New} \times \Delta t \quad (7)$$

and if Δt is adopted as 1 (t is a discrete variable that indicates the iteration number, and Δt indicates the iterative incremental step):

$$X_i^{New} = X_i + V_i^{New} \quad (8)$$

where V_i is called the velocity of particle i , and is defined by the equation:

$$V_i^{New} = \delta(t) \times w_I \times V_i + rnd_1 \times w_M \times (b_i - X_i) + rnd_2 \times w_C \times (b_G - X_i) \quad (9)$$

The dimension of vectors is the number of decision variables. The first term of (9), represents the *inertia* or *habit* of the particle: keeps moving in the direction it had previously moved. $\delta(t)$ is a function decreasing with the progress of iterations, that reduce, progressively, the importance of *inertia* term. The second term represents the *memory*: the particle is attracted to the best point in its trajectory-past life. The third term represents the *cooperation* or *information exchange*: the particle is attracted to the best point found by all particles. The pa-

rameters w_I , w_M and w_C are weights fixed in the beginning of process; rnd_1 and rnd_2 are random numbers sampled from a uniform distribution $U(0, 1)$. The movement of a particle is represented in the Fig. 2. The rule is applied iteratively, until there are no changes in the b_G or the number of iterations reaches a limit.

There are some striking points in Classical PSO, such as: a) it depends of a number of parameters defined externally by the user, and most certainly with values that are problem-dependent. This is certainly true for the definition of weights w_I , w_M and w_C : a delicate work of tuning is necessary in the most of practical problems; b) the external definition of decreasing function, $\delta(t)$, require some caution, because is intuitive that if *inertia* term is eliminated at an early stage of the process, the procedure risks to be trapped at some local minimum; c) last, the random factors $rnd_{1, 2}$, while introducing stochastic flavor, only have a heuristics basis and are not sensitive to evolution of the process. Introducing EPSO metaheuristic it be intend overcome these limitations.

The strategy of PSO, as an algorithm, will be described in the section II.D, where the multi-objective metaheuristic FEPSO, proposed in this work, is explained. This is because FEPSO is an extension of PSO and EPSO.

Evolutionary/Self-Adapting Particle Swarm Optimization (EPSO)

PSO can be observed as a proto-evolutionary process, because exists a mechanism to generate new individuals from a previous set (the movement rule). It is not a explicit selection mechanism in the darwinist sense. However the algorithm exhibits a positive progress rate (evolution) because the movement rule induces such property implicitly. The idea behind EPSO is to grant a PSO scheme with an explicit selection procedure and with self-adapting properties for its parameters. The self-adaptative Evolution Strategies (σ A-ES) model, although there are many variants, may be represented by the following procedure:

- Each individual of generation is duplicated;
- The strategic parameters of each individual are undergo mutation;
- The object parameter of each individual are mutated under a procedure commanded by its strategic parameters (this generates new individuals);

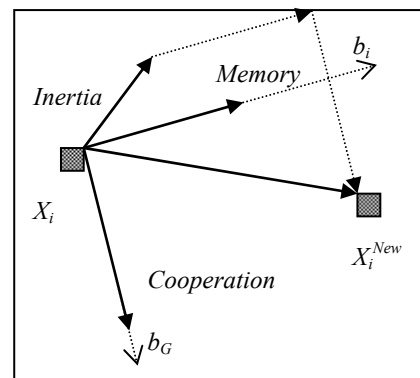


Figure 2 – Movement Rule of PSO in two dimensions

- A number of individuals are undergo recombination (this also generates new individuals);
- From the set of parents and sons (the original and the new individuals), the best fit are selected to form a new generation.

If both strategies (σ SA-ES and PSO) are combined, it is possible to create such scheme (self-adaptative/evolutionary PSO) (Miranda *et al.*, 2008). At a given iteration, consider a set of solutions that can will keep calling “particles”. Then, the general scheme for EPSO is the following:

- **Replication**: each particle is replicated r times;
- **Mutation**: each particle has its weights mutated;
- **Reproduction**: each mutated particle generates an offspring according to the particle movement rule;
- **Evaluation**: each offspring has its fitness evaluated;
- **Selection**: by stochastic tournament, the best particles survive to form a new generation.

Then, the Movement Rule of EPSO is not changed respect PSO, and is valid the Eq. (8). But the new EPSO velocity operator, is expressed by:

$$V_i^{New} = w_{li}^* \times V_i + w_{Mi}^* \times (b_i - X_i) + w_{Ci}^* \times (b_G^* - X_i) \quad (10)$$

The Movement Rule of EPSO, keeps its terms of inertia, memory and cooperation. However, the symbol $*$ indicates that the parameters will undergo mutation:

$$w_{li}^* = w_{li} + \tau \times N(0,1) \quad (11)$$

$$w_{Mi}^* = w_{Mi} + \tau \times N(0,1) \quad (12)$$

$$w_{Ci}^* = w_{Ci} + \tau \times N(0,1) \quad (13)$$

$$b_G^* = b_G + \tau' \times N(0,1) \quad (14)$$

where $N(0, 1)$ is a random variable with Gaussian distribution, mean 0 and variance 1; τ and τ' are learning parameters (either fixed or treated also as strategic parameters and therefore also subject to mutation). The global best b_G is randomly perturbed too.

In the Fig. 3, a new Movement Rule of EPSO, with the perturbed global best, is represented. Notice that the vector associated with the *cooperation* factor does not point to the global optimum, b_G , but to a mutated location, b_G^* .

An option about randomly disturbed best global, is set by the expression:

$$b_G^* = b_G + w_{Gi}^* \times N(0,1) \quad (15)$$

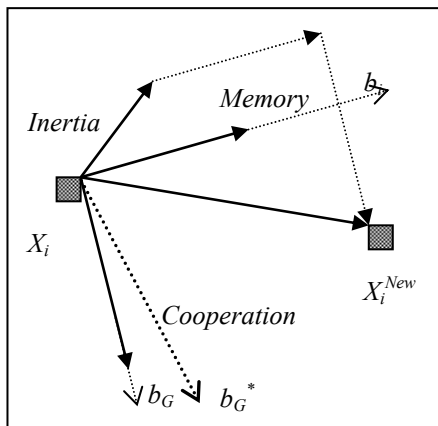


Figure 3 – Movement Rule of EPSO in two dimensions

where w_{Gi}^* is the forth strategic parameter, associated with the particle i . It control the size of neighborhood of b_G where is more likely to find the real global best solution. Another difference respect to the velocity operator of PSO, is that the weights are defined for each particle of swarm (subindex i).

C. Static Fuzzy Decision and Fuzzy Fintness Function

C.1. Fuzzy Decision

The metheuristics SA and PSO were designed, originally, to mono-objective optimizations problems. There are many approaches proposed to extend them to multi-objective optimizations (Smith *et al.*, 2008). In this paper, a new extension capable to treat with no stochastic uncertainties of value is proposed. This kind of uncertainties is present in the preferences between the criterias of multi-objective optimization, and in the satisfaction degree that certain value of a single objective, produce in the decision-maker.

To represent and introduce such uncertainties in the model, the static decision-making in fuzzy environments principle (Bellmand and Zadeh, 1970) is proposed. It is expressed as follows:

Consider a set of fuzzy objectives (whose uncertainties of value are represented by mean of fuzzy sets) $\{O\} = \{O_1, O_2, \dots, O_n\}$ whose membership functions are μ_{Oj} , with $j=1..n$, and a set of fuzzy constraints (whose uncertainties of value in the upper and lower limits, are represented by mean of fuzzy sets) $\{R\} = \{R_1, R_2, \dots, R_h\}$ whose membership functions are μ_{Ri} , with $i=1..h$. The the Decision fuzzy set, results:

$$D = O_1 <C> O_2 <C> \dots <C> O_n <C> R_1 <C> R_2 <C> \dots <C> R_h \quad (16)$$

where $<C>$ is a fuzzy sets operator called “confluence”. The most common confluence, is the intersection. Then, the membership function of D is expressed as:

$$\mu_D = \mu_{O1} C \mu_{O2} C \dots C \mu_{On} C \mu_{R1} C \mu_{R2} C \dots C \mu_{Rh} \quad (17)$$

where C is an opertator (called, in general, t-norm) between values of membership functions. For example, if the confluence is $<C> \equiv \cap$ (intersection), then C is the t -norm $\equiv \min$ (minimum value, for certain instance, of all membership function in eq. (17)). Then, the Maximizing Decision over a set of alternatives, $[X]$, is:

$$\mu_D^{Max} = MAX^{[X]} \{ \mu_{O1} C \mu_{O2} C \dots C \mu_{On} C \mu_{R1} C \mu_{R2} C \dots C \mu_{Rh} \} \quad (18)$$

A t-norm is defined as follows:

If $t: [0, 1] \rightarrow [0, 1]$ is a t-norm, then: a) $t(0,0) = 0$; $t(x,1) = x$; b) $t(x,y) = t(y,x)$; c) if $x \leq \alpha$ e $y \leq \beta \Rightarrow t(x,y) \leq t(\alpha,\beta)$; and d) $t(t(x,y),z) = t(x,t(y,z))$.

Notice that all fuzzy sets (Objectives and Constraints) are “mapping” in the same fuzzy set of decision, D , and are treated the same way.

This type of fuzzy decision is static. It is evaluated in certain instance of occurrences of values corresponding to membership’s functions.

C.2. Fuzzy Sets of Optimization Criteria in the Phase Balancing Problem

To define a multi-objective fuzzy function, will be used these concepts. The development of expressions will be oriented to the objectives and constraints (criteria) of the Phase Balancing problem, but it could be extended to any set of criteria's, represented by fuzzy sets.

Will be assume that the LV feeders system is under operation and exhibit a significant unbalance degree. Four criteria/objectives are introduced in the optimization, and all of them must be minimized: $Loss_T$, $I(\Delta u)$, $|I_{[0]}|_f$, from model in the eq. (1), and NC that represent the number of phase changes (swapping) respect to the reference or existing system A change has associated a cost (and it can disturb the normal service). The constraints will be considered as crisp sets, and any solution that no satisfies Eq. (2)-(4), will be discarded.

All membership functions of fuzzy sets will be construct as linear functions and, then, will be affected by exponentials weights, that represent the importance between the preferences of criteria.

Consider two limits values in a given criteria m : $vMax$ and $vMin$, and let p_μ^m the exponential weight associated to corresponding fuzzy set on vm . Then the membership function to criteria m , is expressed by:

$$\mu_m = 1 ; \text{ if } vMin_m \geq vm \quad (19)$$

$$\mu_m = \left[\frac{vMax_m - vm}{vMax_m - vMin_m} \right]^{p_\mu^m} ; \text{ if } vMin_m \leq vm \leq vMax_m$$

$(p_\mu^m > 1 \rightarrow \text{more importance}; p_\mu^m < 1 \rightarrow \text{less importance})$ (20)

$$\mu_m = 0 ; \text{ if } vMax_m \leq vm \quad (21)$$

In this work, such limits values are obtained as follows: a) the $vMin_m$ will be the result of a PSO mono-objective simulation (that minimize the criteria which variable is vm) with a deterministic fitness function; b) $vMax_m$ is a value depending on the criteria under analysis, as is explained below.

The following calculation of limits values, will correspond to the four criterias considered in the specific Phase Balancing problem.

1) Total Active Power Loss ($Loss_T$)

In this case, a mono-objective PSO is simulated to obtain the minimal power loss of LV feeders system, $vMin_{Loss}$.

The value $vMax_{Loss}$ is obtained by a simulation of a three-phase load flow on reference feeders system.

2) Drop Voltage Index ($I(\Delta u)$)

A LV feeders system is the radial operation. This mean that one option to evaluate the maximum voltage drop, is from voltages in the terminal nodes of each feeder of system. Then, setting two parameters: $uint$ (voltage in tolerance) and $uoutt$ (voltage out of tolerance) applied to the terminal nodes (worse situation of voltage drops), and assigning pertinent values per unit of nominal voltage (for example: $uint = 0.95 [pu]$ and $uoutt = 0.92$), is possible to define the limits as follows: $vMin_u = 1/uint$

and $vMax_u = 1/uoutt$. For each terminal node of LV feeders system, is considered a membership function (19)-(21), with this limits. Then, if nt is the number of terminal nodes, it will be $\mu(vu)_1, \mu(vu)_2, \dots, \mu(vu)_m$ membership functions vinculated to the voltage drops in the feeders system. From this, it proposal the index $I(\Delta u)$ expressed by the geometric mean:

$$I(\Delta u) = \mu(vu) = \sqrt[nt]{\prod_{i=1}^{nt} \mu(vu)_i} \quad (22)$$

The voltages at terminal nodes, in a given instance, results of a three-phase load flow simulation.

3) Homopolar Component Current $|I_{[0]}|_f$

In this case, a mono-objective PSO is simulated to obtain the minimal $|I_{[0]}|_f$ in LV feeders system, $vMin_{|I_{[0]}|_f}$.

The value $vMax_{|I_{[0]}|_f}$ is obtained by a simulation of a three-phase load flow in the reference system.

4) Number of Phase Changes (Swapping) NC

To determine $vMax_{NC}$, is proposed the expression:

$$vMax_{NC} = MAX \{ NC_{PSO_{LossT}}, NC_{PSO_{I(\Delta u)}}, NC_{PSO_{|I_{[0]}|_f}} \} \quad (23)$$

that is the maximum obtained in each PSO simulation. By analogy, to determine $vMin_{NC}$ is proposed:

$$vMin_{NC} = MIN \{ NC_{PSO_{LossT}}, NC_{PSO_{I(\Delta u)}}, NC_{PSO_{|I_{[0]}|_f}} \} - NC0 \quad (24)$$

where $NC0$ is a number externally fixed (it can be 0).

C.2. Fuzzy Fitness Function

The t-norm proposed in this work, is the Einstein Product, defined as:

$$t_{PE} = \frac{x \times y}{2 - (x + y - x \times y)} \quad (25)$$

where x and y are two generic membership functions.

From the properties of a t-norm, presented in section C.1, is possible to construct the membership function of Decision fuzzy set, D , as follows:

$$\mu_D = t_{PE} \{ \mu_{LossT}; I(\Delta u); \mu_{|I_{[0]}|_f}; \mu_{NC} \} = Fff \quad (26)$$

where Fff is the Fuzzy Fitness Function to evaluate the fitness of each individual in the metaheuristic algorithm.

The set of alternatives, $[X]$, for the static fuzzy decision in the Eq. (18), is the set of particles in the swarm of PSO/EPSO, while in SA is each energy state.

This multio-objective approach is valid to the fitness function of any metaheuristic. From this, in the framework of this paper, the metaheuristic SA is extended to FSA an EPSO to FEPSO.

D. Simulation on Real LV Feeders System

The simulation of the two metaheuristics, FSA and FEPSO, is applied on the same real LV feeders system, represented in the Fig. 4, existing in the city of Bariloche, province of Rio Negro, Argentina. It corresponds to one of the six output of a Medium Voltage (MV)-Low Voltage (LV) substation, located in a low-incomes suburban area. For this reason there are only single phase loads in the feeders. This system is adopted as

reference. It can observe the connection of loads to phases $[R]$, $[S]$ and $[T]$. The conductors of feeders have the followings parameters: **Pr**: $3 \times 95 \text{ [mm}^2\text{]}$, ($r = 0.372 + j \cdot xl = 0.0891$) $[\Omega]/[\text{km}]$; **SI, SII, SIII, SIV, SV, SVI, TI, TII, TIII** and **TIV**: $3 \times 35 \text{ [mm}^2\text{]}$, ($r = 1.39 + j \cdot xl = 0.0973$) $[\Omega]/[\text{km}]$. The number of loads is $nL = 115$.

The FSA algorithm follows the procedure described in section B.1, by replacing the Energy function E to Fff .

The FEPSO algorithm, is described in the flow-chart of Fig. 5. $NIterMax$ is the maximum number of iterations externally defined. It possible to observe the scheme corresponding to PSO, by eliminating the processes called Evolutive Operators and MultiObjective. By eliminating only the process called MultiObjective, it observe the scheme corresponding to EPSO.

The parameters used in FSA, are listed below:

- Initial Temperature*: $T_0 = 1.0$;
- Number of Iteration to the same Temperature*: $N_T = 100$;
- Maximum Number of Iterations without improvement of fitness function (stopping condition)*: $nMaxI = 300$;
- Cooling Rate*: $\alpha = 0.8$;
- Function of Generation of Neighborhood*, $q(i, j)$: this identification is making by selection, randomly, of one single phase load, and connecting it in a changed phase.
- k_B Constant (to eq. (6))*: $k_B = 0.00025$.

The Fuzzy fitness function, Fff , is evaluated from the results of a three-phase load flow.

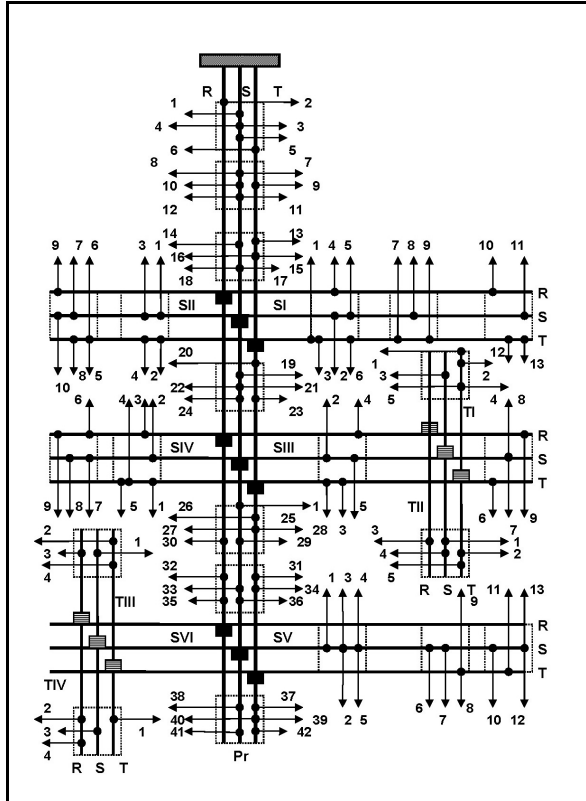


Figure 4 – Real LV Feeders System to Simulation

The Table 1 shows the results corresponding to application of two metaheuristics (FSA y FEPSO). Table 2 shows the complete results of swapping Phase Balancing for mono-objective PSOs and FEPSO. $[S]$ is the loads power vector [kVA] and $[d]$ the nodes distance vector, respect to substation output [km]. The power factor is 0.85.

The model for this application (phase balancing), is static. This implies that the feeders system is analyzed for the worse scenario of the grow load planning, corresponding to the peak of demand for certain period of time (one year, typically). System evolution is not re-

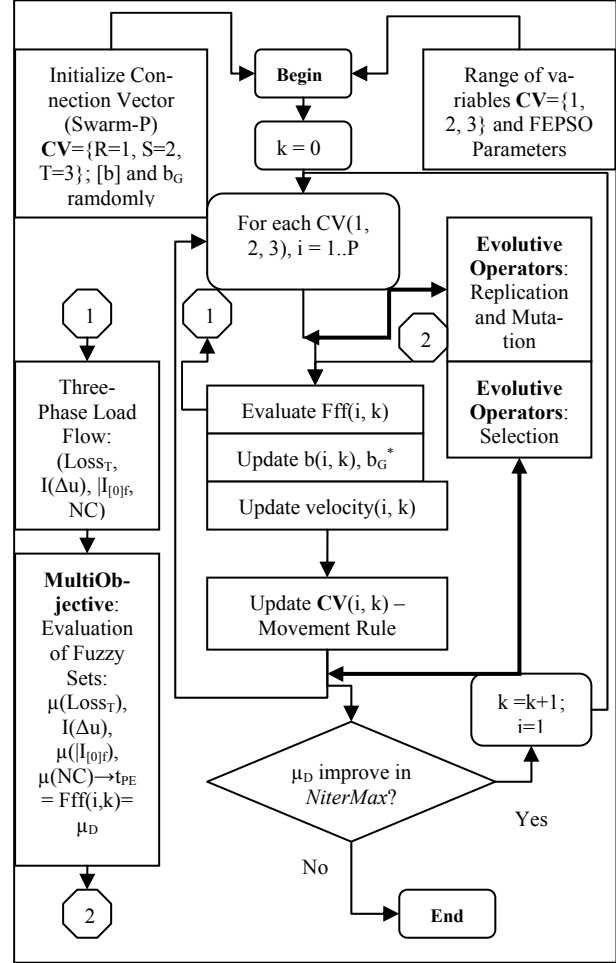


Figure 5 – Flow Chart of FEPSO in Phase Balancing

Table 1: Results of Metaheuristics FEPSO-FSA-PSO

Reference System Results:						
Loss _T [kW] = 13.02 I _{0j} [A] = 47.6 I(Δu) = 0 NC = 0						
PSO	Size	Time[min]	Loss_T	 I_{0j} 	I(Δu)	NC
Loss _T	150	45	6.94	18.93	0.32	81
I _{0j}	150	37	10.16	0.100	0.00	79
I(Δu)	150	43	7.02	13.80	0.34	85
FSA	Size	Time[min]	Loss_T	 I_{0j} 	I(Δu)	NC
t _{PE}	----	63	9.35	1.73	0.11	72
FEPSO	Size	Time[min]	Loss_T	 I_{0j} 	I(Δu)	NC
t _{PE}	200	48	7.21	0.4	0.27	59

Table 2: Complete Swaping Phase Balancing Results of Metaheuristics PSO (three simulations) and FEPSO

<p>Feeder Pr [All elements follows the index order from Fig. 4]</p> <p>[S] = [1.8 1.15 1.15 1.95 1.15 1.15 1.13 1.14 1.15 1.15 1.14 1.15 2.93 1.12 1.13 1 1.15 1.15 1.15 1.15 1.15 1.17 1.16 1.15 1.13 1.13 1.12 1.13 1.15 1.15 1.15 1.18 1.16 1.15 1.15 1.17 1.15 1.18 1.36 1.36 1.36 1.36]</p> <p>[d] = [0.035 0.035 0.035 0.035 0.035 0.035 0.045 0.045 0.045 0.045 0.045 0.045 0.065 0.065 0.065 0.065 0.065 0.065 0.095 0.095 0.095 0.095 0.095 0.095 0.16 0.16 0.16 0.16 0.16 0.16 0.45 0.45 0.45 0.45 0.45 0.45 0.7 0.7 0.7 0.7 0.7]</p> <p>[PSO(Loss_T)] = [T S R S S S S T T S R S R S S S S T S T T R T T T S T R T T T S T R T T R S R R R S]</p> <p>[PSO(I_[0])] = [S S S T S S S R T R R R S R S T T S S R S S S R T R S T R T R S T T T R S R R T R]</p> <p>[PSO(I(Au))] = [R S R T R R T R S R R R S R R R T T T R R S T T R T S T T S T T R R T R S S S S]</p> <p>[FEPSO] = [R R S T R R T R S R R R S R R R T R R R T S R R T S T T R T S R R T S S T T S S]</p>
<p>Feeder SI [All elements follows the index order from Fig. 4]</p> <p>[S] = [1 1.15 1 1.15 1.155 1 1.15 1.17 1.15 1.15 1.18 1.125 1.125]</p> <p>[d] = [0.1 0.1 0.1 0.1 0.1 0.1 0.25 0.25 0.25 0.3 0.3 0.3 0.3]</p> <p>[PSO(Loss_T)] = [T R R T S R R T R T S T R] -- [PSO(I_[0])] = [T T T T S R S R T S R S R]</p> <p>[PSO(I(Au))] = [T T S S R R T S T S S T S] -- [FEPSO] = [T S T R S R T S T R S T T]</p>
<p>Feeder SII [All the elements follows the index order from Fig. 4]</p> <p>[S] = [1.15 1.158 1.125 1.125 1.118 1.125 1 1.15 1.15 1.18]</p> <p>[d] = [0.15 0.15 0.15 0.15 0.3 0.3 0.3 0.3 0.3]</p> <p>[PSO(Loss_T)] = [T T R T T R R T R R] -- [PSO(I_[0])] = [T T T R S T R R S T] -- [PSO(I(Au))] = [S T T T S S R T R] -- [FEPSO] = [R T T T S S S R T R]</p>
<p>Feeder SIII [All the elements follows the index order from Fig. 4]</p> <p>[S] = [1 1.15 1 1.155 1.155 1.18 1.15 1.155 1.15] -- [d] = [0.25 0.25 0.25 0.25 0.35 0.35 0.35 0.35]</p> <p>[PSO(Loss_T)] = [T S R S S T R T R] -- [PSO(I_[0])] = [T T S T T S R T R] -- [PSO(I(Au))] = [R R R S T T S S] -- [FEPSO] = [R S R R T T S S T]</p>
<p>Feeder SIV [All the elements follows the index order from Fig. 4]</p> <p>[S] = [1.15 1.148 1.125 1.125 1.15 1.18 1.125 1 1.15] - [d] = [0.1 0.1 0.1 0.1 0.1 0.2 0.2 0.2 0.2]</p> <p>[PSO(Loss_T)] = [S R R T R T R T T] -- [PSO(I_[0])] = [S R T R R T S T R] -- [PSO(I(Au))] = [T T S S T S T S T] -- [FEPSO] = [T S R R T R S S T]</p>
<p>Feeder SV [All the elements follows the index order from Fig. 4]</p> <p>[S] = [1 1.15 1 1.12 1.125 1 1.18 1.12 1.155 1.15 1.128 1.125 1.125]</p> <p>[d] = [0.1 0.1 0.1 0.1 0.1 0.15 0.15 0.15 0.15 0.2 0.2 0.2 0.2]</p> <p>[PSO(Loss_T)] = [S R T T T R S R R S S S R] -- [PSO(I_[0])] = [R T R R S T T S R S S S R R] -- [PSO(I(Au))] = [R S T T R S T R S S S S] -- [FEPSO] = [R S T T R S T R T S S S S]</p>
<p>Feeder TI [All the elements follows the index order from Fig. 4]</p> <p>[S] = [1 1.15 1 1.128 1.125] - [d] = [0.15 0.15 0.15 0.15 0.15]</p> <p>[PSO(Loss_T)] = [R T T S R] -- [PSO(I_[0])] = [T S R S T] -- [PSO(I(Au))] = [S T T S S] -- [FEPSO] = [T T S S T]</p>
<p>Feeder TII [All the elements follows the index order from Fig. 4]</p> <p>[S] = [1 1.122 1 1.124 1.15] - [d] = [0.095 0.095 0.095 0.095 0.095]</p> <p>[PSO(Loss_T)] = [R T R T R] -- [PSO(I_[0])] = [S T S R T] -- [fUft] = [T T S S T] -- [FEPSO] = [T S S S T]</p>
<p>Feeder TIII [All the elements follows the index order from Fig. 4]</p> <p>[S] = [1 1.152 1.123 1.125] - [d] = [0.135 0.135 0.135 0.135]</p> <p>[PSO(Loss_T)] = [R S S R] -- [PSO(I_[0])] = [R T R R] -- [PSO(I(Au))] = [T R S S] -- [FEPSO] = [S R S T]</p>
<p>Feeder TIV [All the elements follows the index order from Fig. 4]</p> <p>[S] = [1.12 1.15 1 1.8] - [d] = [0.125 0.125 0.125 0.125]</p> <p>[PSO(Loss_T)] = [S R R R] -- [PSO(I_[0])] = [T S R S] -- [PSO(I(Au))] = [T S R S] -- [FEPSO] = [T S S S]</p>

quired, because is not a control model. For this propose (static planning) is introduced, in the studies of the electrical distribution systems, a parameter called *simultaneity factor* of loads. It represents the simultaneous load in the peak of demand. It was considered, in the example, as $fs = 0.6$. This mean that each value $|P + j \times Q|$ (complex power) in [S] vector, Table 2, was multiplied for fs .

The exponential weights can be obtained from the preference matrix between the optimization criteria (Schweickardt and Miranda, 2009). These preferences are certainly subjective, and are expressed in order to Analytical Hierarchy Processes method, proposed by Saaty (1977). Alternatively, these weights can be defined without any previous process, satisfying or not the condition of summatory equal to number of criterias. Follows this way, with an arbitrary choice to emphasize the uncertainties of value, the exponential weights resulting in: $p\mu(Loss_T) = p\mu(I_{[0]}) = p\mu(NC) = 3$; $p\mu(vu) = 4$.

The reference values to form the membership functions to each fuzzy optimization criteria, were results:

$$[Loss_T Min = 6.94 [kW]; Loss_T Max = 13.02 [kW]];$$

$$[I_{[0]} Min = 0.1 [A]; I_{[0]} Max = 47.6 [A]];$$

$$[NC Min = 45; NC Max = 85], \text{ with } NC0 = 34;$$

The parameters used in FPSO, are listed below:

a) *Initial Weights*: $w_I = 0.5$; $w_M = w_C = 2$. In PSO are constants;

b) τ and $\tau' = 0.2$;

c) *Number of Replication for each particle*: $r = 5$;

d) *Maximum Number of Iterations without improvement of fitness function (stopping condition)*: $nIterMax = 400$;

It can observe in the Table 1, the best results reached for the metaheuristic FEPSO, respect to FSA. There are some reason for this: a) FSA exhibit a poor ability to "escape" from local optimal (worse that PSO), when the search space is discrete and the good solutions are very dispersed. In fact, a *bootstrapping* procedure was necessary to change the membership function of $I(Au)$ because this index is *strict*, and the algorithm FSA reached the stopping condition with $F_{ff} = 0$. The *bootstrap*, begins with another more flexible membership function, expressed as $I(Au) = \mu(vu)^{\#} = e^{-[\xi \times Nnot]}$; $0 < \xi \leq 1$, where $Nnot$ is the number of terminal nodes with *out* of tolerance voltage. If, in certain iteration, some solution that satisfy the Eq. (22) is reached, then $I(Au) = \mu(vu)$ (Eq. 22); b) It is no necessary in EPSO, because the self-adaptation introduced by the evaluative operators, allows a self-tuning of weights in the velocity operator. Consequently, this avoid that algorithm to be trapped at some local minimum, or, even worse, at fitness 0; c) In the FSA, even when a *bootstrap* procedure was introduced, after several simulations, always was reached a local minimum. The solution FSA shown in Table 2, is the best reached in 22 executions of algorithm.

As an example of the evolution of minimized variables (objectives, in the optimization problem), in the Fig. 6 a plot of $Loss_T$ vs. time computing, t , for both Metaheuristics, FEPSO and FSA, is presented. $Loss_T$ is the most important parameter to optimize in the classic formulation of phase balancing problem.

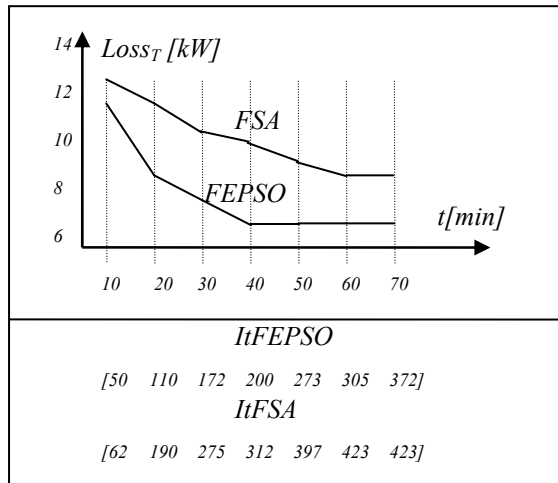


Figure 6 – Time and Iterations in FEPSO and FSA Simulations for Loss Optimization

Additionally, the iterations numbers It_{FEPSO} and It_{FSA} are shown at the observation times in the interval [10, 70] [min], until the convergence of two corresponding algorithms is reached. Very similar plots, can be obtained for the other three objectives: $I_{[0]t}$, $I(\Delta u)$ and NC.

The number of iterations for both metaheuristics, has not significance in a comparative context, because the algorithms's structures are very different.

No "pathological" cases were observed, in which the convergence of both metaheuristics algorithms might be impossible for this application.

As computational cost, (see Table 1) the time computing, $t=T$, to reach the convergence, is the most representative parameter.

Respect of inherent uncertainties in the loads, the model has considered only the worse scenario of grow demand. But a collection of scenarios of grow demand in a given planning horizon, can be integrated in the analysis of the distribution system, with the aim to define different networks topologies in the LV feeders system. Then, a deterministic simulation for each scenario of grow demand-network topology, is performed. Another way to treat with this kind of uncertainties, can be made by mean of fuzzy sets, considering, as separated fuzzy-variables, the active and reactive power in each load connected to the feeders system (Schweickardt and Miranda, 2007).

III. CONCLUSIONS

The paper offers the following contributions:

- A different metaheuristic approach, based in a variant of PSO Metaheuristics, called FEPSO, that produce very good results in multi-objective combinatorial optimization problems, such as Phase Balancing with only single phase loads in a LV feeders system (and four objectives). It is not possible to solve this problem with mathematical programming techniques.
- A comparison FEPSO vs FSA metaheuristics: this allow to observe the advantages of swarm self-adaptive approach. The swarm intelligence prin-

ciples, such as cooperation, combined with evolution strategies, seem like and address of metaheuristics toward solution for any combinatorial optimization problem.

- A method to capture and model the uncertainties of value in the optimization criteria, at the same time that are extends to multi-objective decision making, introducing the fuzzy sets modeling, and fuzzy function fitness.

REFERENCES

- Bellman, R. and L. Zadeh, "Decision-Making in a Fuzzy Environment," *Management Science*, **17**, B-141-164 (1970).
- Kennedy, J. and R. Eberhart, "Particle Swarm Optimization," *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Perth, Australia, 1942-1948 (1995).
- Kirkpatrick, S., C. Gellat and M. Vecchi, "Optimization by Simulated Annealing," *Science*, **220**, 671-680 (1995).
- Metropolis, N., A. Rosenbluth, A. Teller and E. Teller, "Equation of state calculation by fast computing machines," *Journal of Chemical Physics*, **21**, 1087-1092 (1953).
- Miranda, V., H. Keko and A. Duque Jaramillo, "EPSO: Evolutionary Particle Swarms," In *Advances in Evolutionary Computing for System Design*, L. Jain, V. Palade and D. Srinivasan (Eds.), Springer series in Computational Intelligence, **66**, 139-168, (2008).
- Saaty, T., "A Scaling Method for Priorities in Hierarchical Structures," *Journal of Mathematical Psychology*, **15**, 234-281 (1977).
- Schweickardt, G. and V. Miranda, "A Fuzzy Dynamic Programming Approach for Evaluation of Expansion Distributions Cost in Uncertainty Environments," *Latin American Applied Research*, **37**, 227-234 (2007).
- Schweickardt, G. and V. Miranda, "A Two-Stage Planning and Control Model Toward Economically Adapted Power Distribution Systems using Analytical Hierarchy Processes and Fuzzy Optimization," *International Journal of Electrical Power & Energy Systems*, **31**, 277-284 (2009).
- Smith K., Everson J., Fieldsend R., Misra R., and Murphy C., "Dominance-based Multi-objective Simulated Annealing," *IEEE Transactions on Evolutionary Computation*, **12**, pp. 323-342, (2008).
- Zhu, J., B. Griff and M. Chow, "Phase Balancing Using Mixed-Integer Programming," *IEEE Trans. Power Systems*, **13**, 465-472 (1998).

Received: June 16, 2009.

Accepted: May 11, 2010.

Recommended by Subject Editor José Guivant.